# Project 2: Photo Gallery Application - SQL and NoSQL Variants

**References:**

[1] A. Bahga, V. Madisetti, “Cloud Computing Solutions Architect: A Hands-On Approach”, ISBN: 978-0996025591

[2] <https://aws.amazon.com/documentation/>

**Due Date:**

The project report will be **due in 2 weeks from the date assigned.**

This project is about creating a Photo Gallery application that allows users to upload photos, view details of photos, and search photos. The application is implemented in Python and uses the Flask web framework. The Flask application is deployed on an Amazon EC2 instance. Photos uploaded to the application are stored in an Amazon S3 bucket. In the SQL variant of the application, the records of photos are maintained in a MySQL database instance on Amazon RDS. Whereas in the NoSQL variant of the application, the records of photos are maintained in a DynamoDB table.

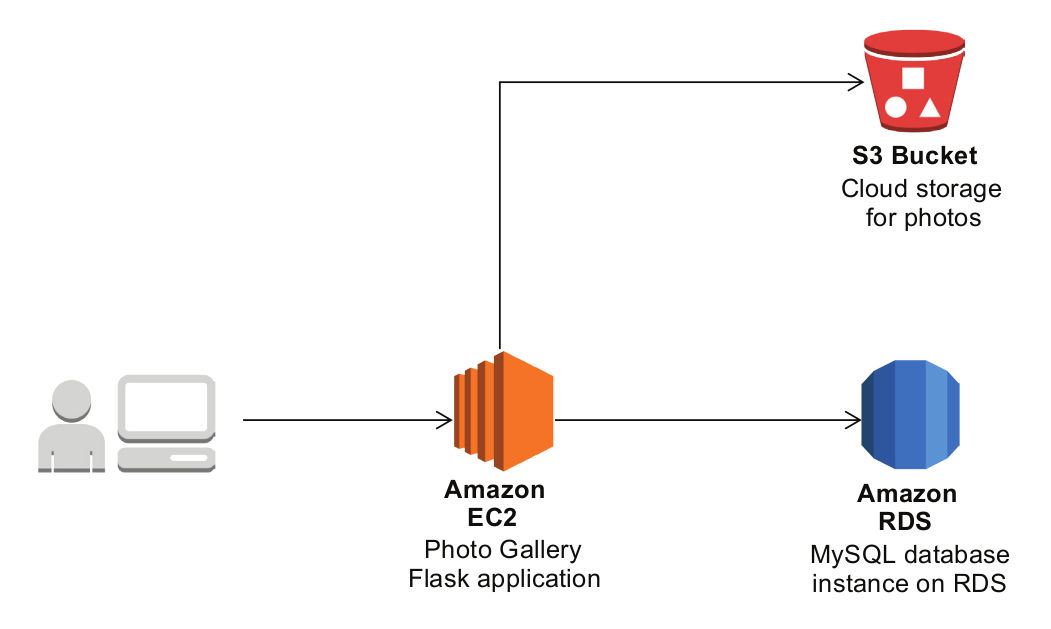


Fig. Architecture diagram of the Photo Gallery application - SQL Variant

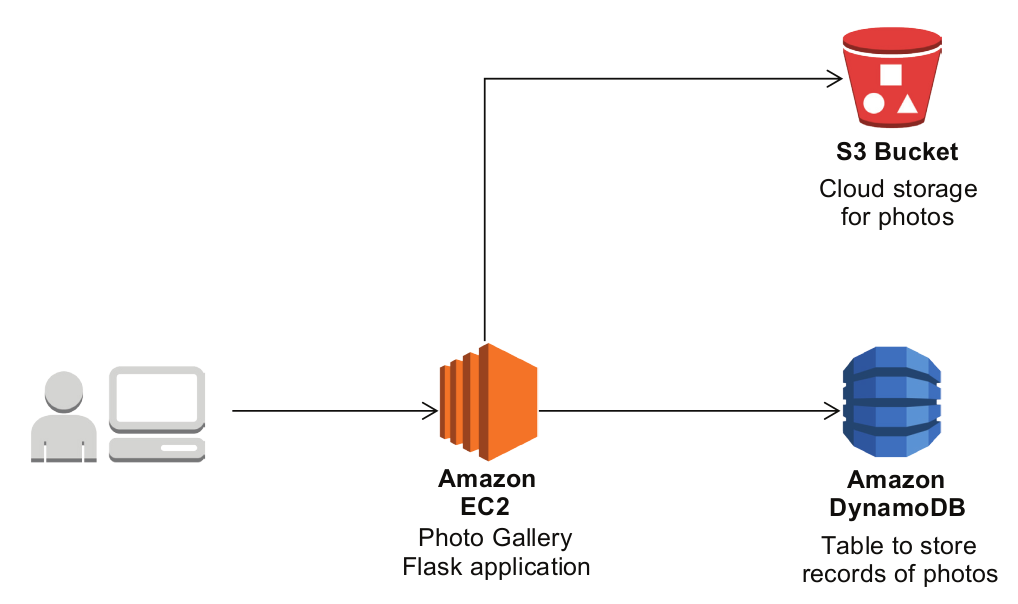


Fig. Architecture diagram of the Photo Gallery application - NoSQL Variant

Follow the steps below to setup the Photo Gallery application in your AWS account.

**1. Create an S3 Bucket for hosting the static website for the application**

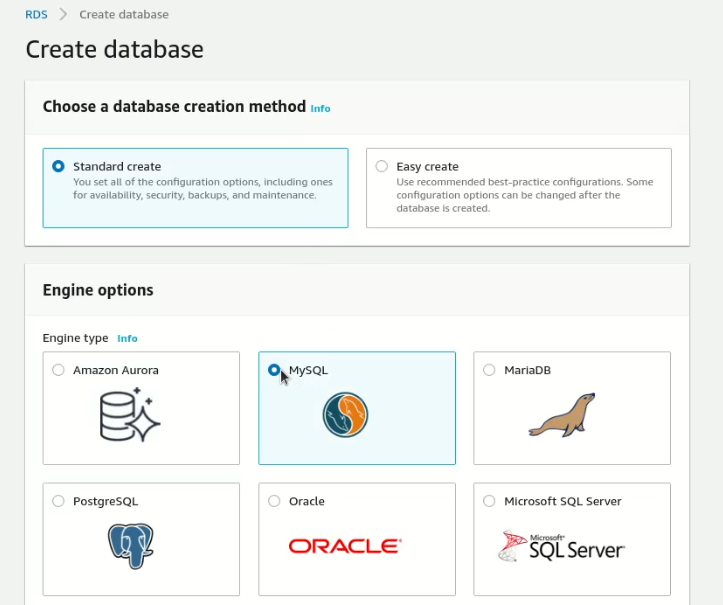
* Create a new S3 bucket for hosting the static website and enable static website hosting for the bucket.

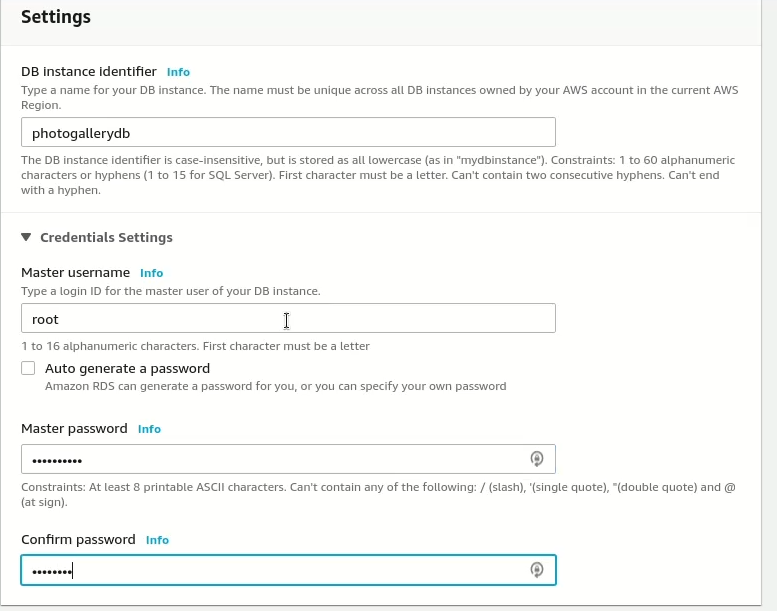
**2. Create an S3 Bucket for storing photos**

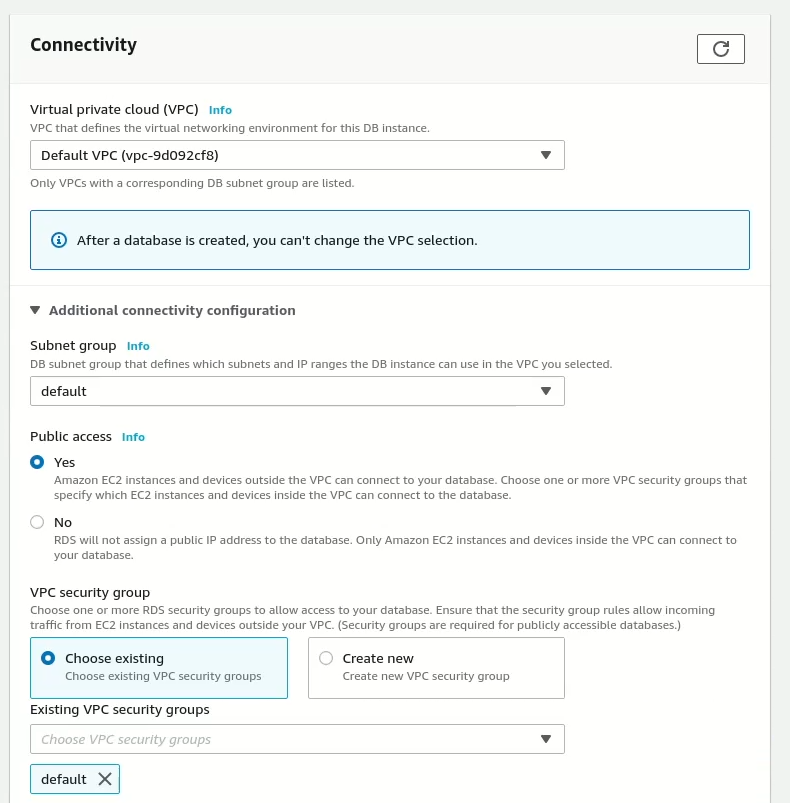
* Create a new S3 bucket for storing photos.
* Create a folder named ‘photos’ in this bucket.
* Add a bucket policy below to enable public access to the photos uploaded. Replace ‘mybucketname’ with the name of the S3 bucket created.

|  |
| --- |
| {  "Version": "2012-10-17",  "Id": "Policy1538026169421",  "Statement": [  {  "Sid": "Stmt1538026165732",  "Effect": "Allow",  "Principal": "\*",  "Action": "s3:GetObject",  "Resource": "arn:aws:s3:::mybucketname/photos/\*"  }  ]  } |

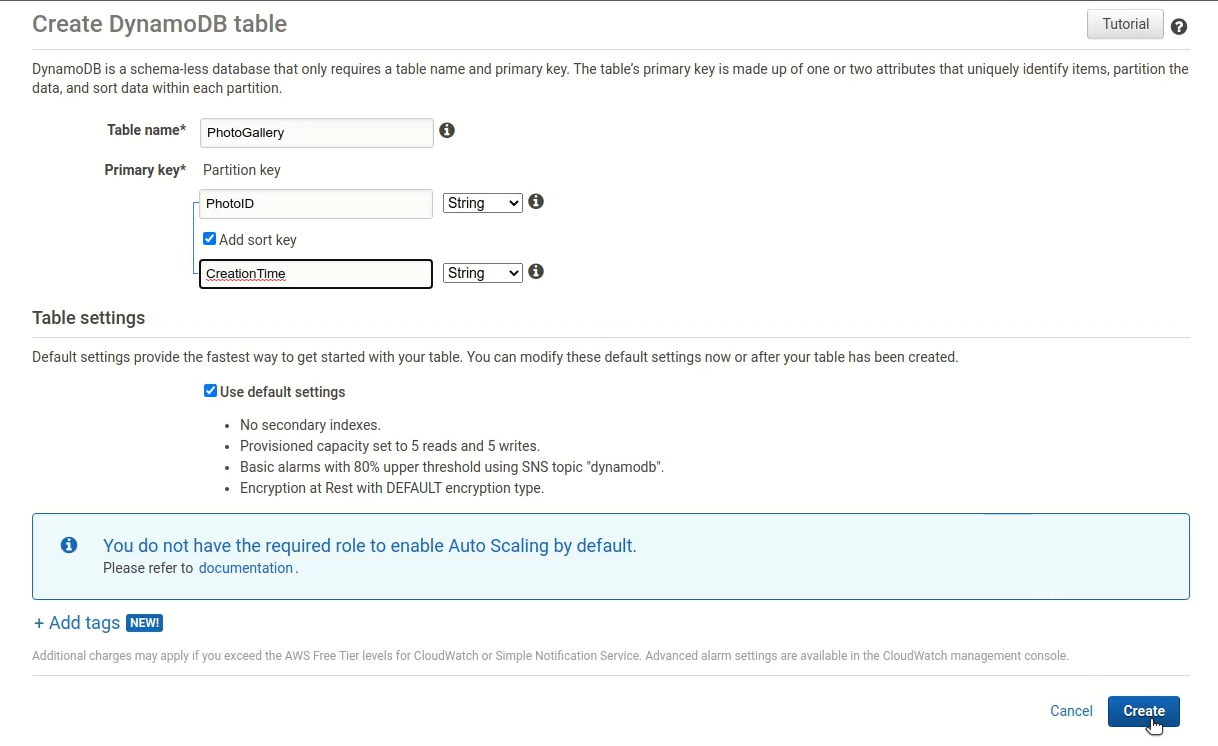
1. **Create an RDS database instance for storing records of photos**

****

****

****

1. **Create a DynamoDB table for storing records of photos**

****

**5. Setup and run the Photo Gallery application**

Setup and run the photo gallery application, as shown in demo video. Add a more photos and try browsing and searching for photos.

**Create an EC2 Instance to deploy the Flask apps:** a) First create a key-pair b) Save the private key file c) then create the EC2 instance with that key-pair

**Create key pair:** navigate to EC2 in AWS Management Console->select key pairs (from Network & Security)->click Create key pair button->Provide a name->select RSA and PEM->click Create Key Pair button->a PEM file will be created in the download folder with the PRIVATE KEY. Save this PEM file.

**Create EC2 Instance:** navigate to EC2 in AWS Management Console->Launch instances->provide a name->select Windows->select the key-pair created earlier->create security group (check Allow HTTPS traffic from the internet)->click Launch Instance button->Click on Instance link->Connect->RDP Client->Download Remote Desktop File (SAVE THIS FILE)->Username: Administrator->click Get Password->Upload private key file (PEM file saved as part of creating key-pair file) to decrypt password. Click on RDP file (saved to download folder) and use this username and password to connect to the instance. You will have to click on “Download remote desktop file” (screenshot below) every time to connect (with the same password though):

A screenshot of a computer

Description automatically generated

\*Copy the PROJECT 2 folder (from D2L) to the EC2 instance C: drive. Install PyCharm IDE. Run the createtable.py and app.py scripts in a new Python project (make sure to install ExifRead, Flask, boto3 and PyMySQL packages). Remember to copy the “templates”, “media”, “static” folders to the python project. Alternatively, in COMMAND prompt (Run as ADMIN) cd C:\COSC40233-Cloud\_Computing\_Labs\_2022\Project 2 Solution\Project2-files\lab2-files\SQL and then follow the video i.e.

pip3 install exifread flask PyMySQL boto3 mysql

python3 createtable.py

python3 app.py

(CTL + C) to quit

**VVI:** The app URL (localhost:5000) only works when the app.py script is running. Also, the URL only works in the local EC2 instance. Also, change the port number to 5001 (in 2 places) in the Python script to run the app if you have issues with port 5000.

**VVI: RDS->Click on DB Name (i.e. photogallerydb)->click on security group name (under Security)->click on security group link->Action->Edit Inbound rule->Add an Inbound rule as below:**

A screenshot of a computer

Description automatically generated

**PROJECT 2 Submissions:**

1. Watch the video and complete the steps (except 2 steps below) – 40%
2. Complete the view\_photo(photoID) function in the app.py file (SQL Variant) – 10%
3. Complete the view\_photo(photoID) function in the app.py file (NoSQL Variant) – 10%

def view\_photo(photoID):

1. Write a detailed report (at least 8 to 10 pages excluding the cover page) about your experiences with this project. You may include the design, architecture, explanation of code snippets, learning outcome, challenges, opportunities among other things. There should be a cover page as well. 30%
2. Submit the URLs to your photo gallery web applications (SQL variant and NoSQL variant). 10%